

# Theoretical Foundations of CP-based Lagrangian Relaxation <sup>\*</sup>

Meinolf Sellmann

Cornell University  
Department of Computer Science  
4130 Upson Hall  
Ithaca, NY 14853  
sello@cs.cornell.edu

**Abstract.** CP-based Lagrangian Relaxation allows us to reason on local substructures while maintaining a global view on an entire optimization problem. While the idea of cost-based filtering with respect to systematically changing objective functions has been around for more than three years now, so far some important observations have not been explained. In this paper, we prove a simple theorem that explains a variety of effects that are encountered in practice, the most counter-intuitive being the fact that suboptimal Lagrangian multipliers can have stronger filtering abilities than optimal ones.

**Keywords:** cost-based filtering, optimization constraints, relaxed consistency

## 1 Introduction

When analyzing and modeling real-world optimization problems, we often find that they consist in a conglomerate of much simpler substructures. These substructures often exhibit special properties that do not hold for the overall problem, frequently they have been studied before, and efficient algorithms for their optimization exist. When solving a composed problem, we would like to exploit our knowledge of the substructures that constitute the problem as a whole. In constraint programming (CP), the core idea is to encapsulate each substructure in a constraint, whereby traditionally we treat the objective function independently of the other constraints (we formulate our search for an improving solution as a feasibility problem).

In order to improve the filtering abilities of our constraint program, we may wish to follow the successful trend to develop rather strong global constraints and consider to combine the objective function with the corresponding constraints that define each substructure. That is, for each substructure we formulate an *optimization constraint* [9] that expresses our wish that we are looking for improving solutions that are at least feasible when we ignore all other substructures.

---

<sup>\*</sup> This work was supported by the Intelligent Information Systems Institute, Cornell University (AFOSR grant F49620-01-1-0076).

Then, by exploiting our knowledge on the individual substructures, we can develop efficient *cost-based filtering* algorithms [9] that we use for domain filtering in each choice point. That is, by locally reasoning about one optimization substructure at a time, we try to simplify the overall problem by eliminating all variable assignments that will not yield to feasible and improving solutions with respect to that optimization constraint. As it is standard in CP, this process is iterated until we reach a fix point where no optimization constraint alone is able to shrink the domain of the variables further.

There is one problem with this procedure though: By looking at only one substructure at a time, we will usually vastly underestimate the objective costs (that we try to minimize) that can still be achieved. Consequently, cost-based filtering will be rather ineffective since most often we will assume that it is still possible to improve the objective even when this is not the case. In order to obtain a good estimate on the best objective performance that is still achievable, we simply cannot afford to ignore the other substructures completely. Instead, we need to introduce some kind of global view on the entire problem that allows us to compute a more accurate bound estimate.

At first, it sounds as if this necessity stood in contradiction with the idea of arguing locally about one substructure at a time. As it turns out, though, for linear problems there is an easy way to take the other constraints into account while still reasoning about individual substructures only. The simple idea consists in softening the other constraints and placing them in the objective function. Via a penalty term we increase the objective's cost, the more violated the other constraints are the higher the price that has to be paid. In mathematical programming, this softening and penalizing of constraints is known as *Lagrangian relaxation*.

For the purpose of cost-based filtering, Lagrangian relaxation allows us to consider a global bound on the overall problem while exploiting local structure. Following the work in [8] where local reasoning was combined with strengthened lower bounds by adding cuts in a Lagrangian fashion, the idea of CP-based Lagrangian Relaxation was first formalized in [23] and later used in [24, 22, 25] for the automatic recording problem that consists in a conjunction of a Knapsack and a weighted stable set problem. While numerous practical applications of this procedure have shown that CP-based Lagrangian Relaxation can boost the practical performance of solution approaches (see [25, 26, 19] for examples), there are many outstanding theoretical questions: Why should one perform filtering for sub-optimal Lagrangian multipliers? What are the best penalties for filtering? Do we need filtering algorithms for all substructures, or can filtering be as effective when we only use one filtering algorithm for just one substructure? And how effective can CP-based Lagrangian Relaxation be in the best case?

In this paper, we prove a simple but central theorem that answers these questions. Our hope is, that the knowledge of the underlying theory of CP-based Lagrangian Relaxation will help to focus on the important issues when working on practical solution approaches. The paper is organized as follows: In the next section, we briefly review the concept of CP-based Lagrangian Relaxation. After

giving an example in Section 3, we investigate the theoretical properties of the method in Section 4. A practical experiment presented in Section 5 shows the behavior of CP-based Lagrangian relaxation for a real-world problem instance. Finally, we conclude in Section 6.

## 2 What is CP-based Lagrangian Relaxation?

Given a natural number  $n$  and vectors  $l, u \in \mathbb{N}^n$ , we consider an integer linear optimization problem (IP) consisting of the two constraint families  $\mathcal{A}: Ax \leq b$ ,  $x_i \in \{l_i, \dots, u_i\}$ , and  $\mathcal{B}: Bx \leq d$ ,  $x_i \in \{l_i, \dots, u_i\}$ :

$$\begin{array}{ll} \text{Minimize} & L = c^T x \\ \text{subject to} & Ax \leq b \\ & Bx \leq d \\ & x_i \in \{l_i, \dots, u_i\}. \end{array}$$

There exist many examples of real-world problems that can be decomposed naturally in that way. To mention just three: The well known traveling salesman problem for example can be viewed as a combination of an assignment problem and a special spanning tree problem [13, 14]. The automatic recording problem [24] consists in a combination of a Knapsack and a weighted stable set problem. And the network design problem consists in a shortest path problem combined with a special continuous Knapsack problem [26].

### 2.1 Pruning

A common way to achieve a global lower bound  $\bar{L}$  on such a problem is to drop the integrality constraints  $x_i \in \{l_i, \dots, u_i\}$  and to replace them by  $l_i \leq x_i \leq u_i$  instead. We get

$$\begin{array}{ll} \text{Minimize} & \bar{L} = c^T x \\ \text{subject to} & Ax \leq b \\ & Bx \leq d \\ & l \leq x \leq u. \end{array}$$

In mathematical programming, this or a similar bound would be used to identify parts of the search space that do not contain improving solutions, an idea that is called pruning: If we find that all solutions that could potentially be found in the subtree rooted at the current choice point are non-improving, we can backtrack immediately. Of course, our algorithm works the better the more accurate our bound estimate is. The trade-off here consists in finding a rather accurate bound that can still be computed quickly.

### 2.2 Filtering

In CP, we want to do more than pruning and that is: We would like to be able to identify those variable assignments that will not yield to feasible improving

solutions. The process of removing values from variable domains because of such considerations is called *cost-based filtering* [9].

The strongest form of cost-based filtering for an optimization constraint would be to require that *all* variable assignments be identified that will not yield to improving and feasible solutions with respect to the given constraint. This corresponds to achieving generalized arc-consistency for an optimization constraint. However, even for very elementary constraints such as Knapsack constraints or shorter path constraints this turns out to be an NP-hard problem [21], which is why the idea of *relaxed consistency* was introduced in [7].

The idea of relaxed consistency is the following: Considering the  $\bar{L}$ -bound, we could use that bound for cost-based filtering by solving a series of LPs  $\bar{L}[x_i = v]$  where we set some variable  $x_i$ ,  $1 \leq i \leq n$ , to some value  $v \in \{l_i, \dots, u_i\}$ . Then, given an upper bound  $B$  (an upper bound is usually given by the value of the best known solution computed so far, but in some approaches, like for instance in the context of CP-based column generation, it can also reflect our search for negative reduced costs), we can eliminate  $v$  from the domain of  $x_i$  if  $\bar{L}[x_i = v] \geq B$ .<sup>1</sup> After filtering in this fashion, we are sure that all potential variable assignments are eliminated from the problem that would cause the bound  $\bar{L}$  to exceed the current best known solution value. Then, we say that this procedure achieves a state of relaxed- $\bar{L}$ -consistency [7]. With respect to the lower bound that we have chosen, this is the strongest form of consistency that can be achieved for an optimization constraint.

The problem with the previous probing procedure is that it requires to re-optimize a dual feasible LP many times (and that is the case even if we used a more sophisticated binary search to compute the new bounds on the variables), which is usually unattractive with respect to the required computation time. Therefore, it has been suggested to estimate the loss in performance by carrying out exactly one dual reoptimization step. This method is known as *reduced-cost filtering* [18]. It is computationally cheap, but since it only indirectly exploits the structure of the problem it has a tendency to be rather ineffective.

### 2.3 How to Maintain a Global View While Reasoning Locally

To improve the inherent trade-off between computational effort and effectivity, next we may try to decompose the problem so as to be able to reason about the well-shaped substructures that we identified in the problem. Assume that efficient cost-based filtering algorithms  $\text{Prop}(\mathcal{A})$  and  $\text{Prop}(\mathcal{B})$  exist for the constraint families  $\mathcal{A}$  and  $\mathcal{B}$ , respectively. The obvious approach to solve problem  $L$  exactly is to apply a branch-and-bound algorithm using linear relaxation bounds for pruning and the existing filtering algorithms  $\text{Prop}(\mathcal{A})$  and  $\text{Prop}(\mathcal{B})$  for tightening the problem formulation in every choice point. That is, we can easily use

---

<sup>1</sup> Note that, due to  $\bar{L}[x_i \leq v] \geq \bar{L}[x_i \leq w]$  for all  $w \geq v$  (the lower bound constraints follow analogously), this procedure will not split the domains of the variables  $x$ . That is, after the filtering the domains of the variables  $x_i$  can again be represented as closed intervals:  $x_i \in \{\hat{l}_i, \dots, \hat{u}_i\}$  for some  $\hat{l}_i \geq l_i$  and  $\hat{u}_i \leq u_i$ ,  $1 \leq i \leq n$ .

a global bound for pruning with respect to the objective function, and we know how to use local structure for filtering with respect to local feasibility and optimization improvement.

However, even though  $\text{Prop}(\mathcal{A})$  and  $\text{Prop}(\mathcal{B})$  may be effective for the substructures they have been designed for, their application to the combined problem is usually not. This is because tight bounds on the objective cannot be obtained by taking only a subset of the restrictions into account. An accurate bound on the overall problem can only be computed by looking at the entire problem, but it cannot be achieved by looking at either one constraint family only. The core challenge here is to bring together the merits of reasoning about one well-shaped and well-understood substructure at a time while still maintaining a global view on the overall problem.

A standard method from mathematical programming allows us to do exactly that: Lagrangian relaxation permits us to bring together the advantages of a tight global bound and the existing filtering algorithms that exploit the special structure of their respective constraint families.<sup>2</sup> The idea of Lagrangian relaxation is simple: we soften some of the constraints by removing them as hard constraints and penalizing their violation in the objective function. For this purpose, we introduce a vector of Lagrangian multipliers  $\lambda \leq 0$  (which are our penalty factors) and define the *Lagrangian subproblem*

$$\begin{array}{ll} \text{Minimize} & L_{\mathcal{B}}(\lambda) = c^T x - \lambda^T (Ax - b) \\ \text{subject to} & Bx \leq d \\ & x_i \in \{l_i, \dots, u_i\}. \end{array}$$

For every choice of  $\lambda \leq 0$ , linear programming theory tells us that  $L_{\mathcal{B}}(\lambda)$  is a lower bound on  $L$ . Therefore, we can use this lower bound (for all choices of  $\lambda \leq 0$ !) and filter with respect to every objective function that is generated in that way. This is the core idea of CP-based Lagrangian Relaxation. In order to achieve good lower bounds, we embed this filtering in a search for good penalties: The Lagrangian multiplier problem or *Lagrangian dual* consists in finding the best set of penalties that will maximize the lower bound:

$$\begin{array}{ll} \text{Maximize} & G = L_{\mathcal{B}}(\lambda) \\ \text{subject to} & \lambda \leq 0. \end{array}$$

There exist a variety of iterative algorithms for optimizing the Lagrangian dual which gives rise to a piece-wise linear, concave function. Standard algorithms used in the literature are subgradient algorithms, bundle methods, and the volume algorithm [1, 3, 5, 11, 12]: we start by choosing a set of multipliers  $\lambda \leq 0$ , solve the Lagrangian subproblem  $L_{\mathcal{B}}(\lambda)$  and update the multipliers guided by the solutions of the Lagrangian subproblems that were solved. This process is iterated until the process converges or until we are satisfied with the solution quality.

---

<sup>2</sup> The idea of Lagrangian relaxation was first presented in [6] for resource allocation problems. Held and Karp used it for the TSP [13, 14], and it has been applied in many different areas since then. For a general introduction we refer the reader to [1].

In summary, CP-based Lagrangian Relaxation consists in the following procedure: Assume we are given a linear optimization problem that consists in the conjunction of two constraint families  $\mathcal{A}$  and  $\mathcal{B}$  for which an efficient filtering algorithm  $\text{Prop}(\mathcal{B})$  is known. We try to optimize Lagrangian multipliers for  $\mathcal{A}$  and use  $\text{Prop}(\mathcal{B})$  for filtering in each Lagrangian subproblem  $L_{\mathcal{B}}(\lambda)$ . That is, in each Lagrangian subproblem we use the current lower bound that we obtain to perform cost-based filtering with respect to substructure  $\mathcal{B}$ .

### 3 A Disturbing Example

To make the discussion less abstract, consider the following example that is an instance of the two-dimensional Knapsack problem [17]:

$$\begin{aligned} \text{Minimize} & \quad -3x_1 - 3x_2 - 3x_3 \\ \text{subject to} & \quad 9x_2 - 3x_3 \leq 6 \\ & \quad 9x_1 + 6x_2 - 3x_3 \leq 6 \\ & \quad x_i \in \{0, 1\}. \end{aligned}$$

We view the problem as composed of two Knapsack substructures,  $\mathcal{A}$ :

$$\begin{aligned} \text{Minimize} & \quad -3x_1 - 3x_2 - 3x_3 \\ \text{subject to} & \quad 9x_2 - 3x_3 \leq 6 \\ & \quad x_i \in \{0, 1\}, \end{aligned}$$

and  $\mathcal{B}$ :

$$\begin{aligned} \text{Minimize} & \quad -3x_1 - 3x_2 - 3x_3 \\ \text{subject to} & \quad 9x_1 + 6x_2 - 3x_3 \leq 6 \\ & \quad x_i \in \{0, 1\}. \end{aligned}$$

We assume that a Knapsack filtering algorithm for  $\mathcal{B}$  exists. For our example it makes no difference whether this filtering algorithm actually achieves generalized arc-consistency for the Knapsack constraint [20, 27], or whether it achieves a state of relaxed consistency only [7, 20]. We will present the effect of both types of filtering algorithms. To make the description of our example complete, let us assume that we are looking for solutions with an objective cost lower or equal -3 only.

We observe: Neither with respect to substructure  $\mathcal{A}$  nor with respect to substructure  $\mathcal{B}$  is it possible to filter any value from the variables' domains!

Therefore, following the idea of CP-based Lagrangian Relaxation, we choose a multiplier  $\lambda \leq 0$ , say  $\lambda := -1/2$ , and we relax the substructure  $\mathcal{A}$  into the objective function. We achieve  $L_{\mathcal{B}}(-\frac{1}{2})$ :

$$\begin{aligned} \text{Minimize} & \quad -3x_1 + \frac{3}{2}x_2 - \frac{9}{2}x_3 - 3 \\ \text{subject to} & \quad 9x_1 + 6x_2 - 3x_3 \leq 6 \\ & \quad x_i \in \{0, 1\}. \end{aligned}$$

An optimal integer and fractional solution to this problem is  $x^T = (1, 0, 1)$  with an objective cost of -21/2 which is clearly lower or equal -3. Thus, this lower

bound does not allow us to prune the current node. Moreover, we can see also that cost-based filtering again has no effect (even if we use an arc-consistency filtering algorithm for the Knapsack constraint). However, we are still in the process of finding a good Lagrangian multiplier, so we keep our hopes high and continue with  $\lambda := -1/7$ . We get  $L_{\mathcal{B}}(-\frac{1}{7})$ :

$$\begin{aligned} \text{Minimize} \quad & -3x_1 - \frac{12}{7}x_2 - \frac{24}{7}x_3 - \frac{6}{7} \\ \text{subject to} \quad & 9x_1 + 6x_2 - 3x_3 \leq 6 \\ & x_i \in \{0, 1\}. \end{aligned}$$

The optimal integer and fractional solution to this problem is  $x^T = (1, 0, 1)$  with an objective cost of  $-51/7$  which is also lower or equal  $-3$ . So again, we cannot prune the search at this stage. However, no matter whether we use an arc-consistency filtering routine or whether we only achieve relaxed consistency, now we can infer that  $x_3 = 1$ : The arc-consistency algorithm infers that if  $x_3 = 0$  then the best we can do is to set  $x_2 = 1$  and  $x_1 = 0$  which results in an objective cost of  $-18/7$  which is not lower or equal  $-3$  anymore. In case that we use a filtering routine based on relaxed continuous consistency, the best we can do after setting  $x_3 = 0$  is to set  $x_1 = 2/3$  and  $x_2 = 0$  which results in an objective cost of  $-20/7$  which is also greater than  $-3$ .

At this stage, what we would actually like to do is to remove value 0 from the domain of variable  $x_3$ . However, we have to be very careful here since we are still in the search for an optimal Lagrangian multiplier  $\lambda$ , and the standard approaches for the optimization of the Lagrangian dual are not guaranteed to be robust enough to permit a change of the underlying problem during the optimization. Therefore, we just note that value 0 is subject to deletion of the domain of  $x_3$  right after the Lagrangian dual has been solved.

We may now consider to set  $\lambda = 0$ , which results in  $L_{\mathcal{B}}(0)$ :

$$\begin{aligned} \text{Minimize} \quad & -3x_1 - 3x_2 - 3x_3 \\ \text{subject to} \quad & 9x_1 + 6x_2 - 3x_3 \leq 6 \\ & x_i \in \{0, 1\}. \end{aligned}$$

The best integer solution for this problem is again  $x^T = (1, 0, 1)$  with an objective cost of  $-6$  (the best fractional solution is  $x^T = (\frac{1}{3}, 1, 1)$  with an associated cost of  $-7$ ). But wait:  $L_{\mathcal{B}}(0)$  is the same as  $L_{\mathcal{B}}$ , and for  $L_{\mathcal{B}}$  we had found before that we cannot filter any value from the variables' domains. So how can it be that for  $L_{\mathcal{B}}(-\frac{1}{7})$  we find a lower bound of  $-51/7$ , but are able to detect that  $x_3 \neq 0$ , while for  $L_{\mathcal{B}}(0)$  we find an improved (as a matter of fact: optimal) lower bound of  $-6$  (or  $-7$ ) but cannot filter at all?!

## 4 Properties of CP-Based Lagrangian Relaxation

### 4.1 The Theorem of Filtering Penalties

Let us state the situation that we find ourselves in again: We have found an example that shows that there exist Lagrangian multipliers that yield to a worse bound and at the same time to more filtering. We are left with a heap of questions: How can that be when it is the bound estimate that is the basis of our reasoning that a certain variable assignment will not yield to an improving solution? What are good multipliers for filtering if it is not the same set that solves the Lagrangian dual? How effective can CP-based Lagrangian Relaxation be when compared with relaxed- $\bar{L}$ -consistency? And wouldn't it strengthen our filtering method if we could also filter with respect to substructure  $\mathcal{A}$  by exploiting the cost-based filtering algorithm  $\text{Prop}(\mathcal{A})$ ? In order to answer these questions, we prove the following simple but central *Theorem of Filtering Penalties*:

**Theorem 1.** *Given  $1 \leq j \leq n$ , a value  $v \in \{l_j, \dots, u_j\}$ , let  $L_{\mathcal{B}}(\lambda)[x_j = v]$  denote the IP that evolves when adding the constraint  $x_j = v$  to  $L_{\mathcal{B}}(\lambda)$ . Furthermore, let  $B \in \mathbb{Q}$  denote an upper bound on the objective of  $L$  such that  $\bar{L}[x_j = v] \geq B$ . Finally, denote the continuous relaxation of  $L_{\mathcal{B}}(\lambda)$  by  $\bar{L}_{\mathcal{B}}(\lambda)$ . Then there exists a vector  $\lambda \leq 0$  such that  $L_{\mathcal{B}}(\lambda)[x_j = v] \geq \bar{L}_{\mathcal{B}}(\lambda)[x_j = v] \geq B$ .*

*Proof.* Let  $\lambda \leq 0$  denote a vector of optimal dual values of the constraint family  $\mathcal{A}$  in  $\bar{L}[x_j = v]$ . The theory of Lagrangian relaxation shows that the vector  $\lambda$  defines optimal Lagrangian multipliers for  $\bar{L}_{\mathcal{B}}(\lambda)[x_j = v]$ , and it holds  $\bar{L}_{\mathcal{B}}(\lambda)[x_j = v] = \bar{L}[x_j = v]$ . And therefore,

$$L_{\mathcal{B}}(\lambda)[x_j = v] \geq \bar{L}_{\mathcal{B}}(\lambda)[x_j = v] = \bar{L}[x_j = v] \geq B. \quad (1)$$

□

While this theorem looks rather technical, it has a set of important implications that will answer all our questions above:

*Remark 1.* The Filtering Penalties Theorem shows that for every variable  $x_j$  and value  $v \in \{l_j, \dots, u_j\}$  that can be filtered with respect to the relaxation  $\bar{L}$ , there exists a vector of Lagrangian multipliers that allows us to filter this value with respect to the constraint family  $\mathcal{B}$ . More precisely, this can already be achieved when the filtering algorithm for  $\mathcal{B}$  only achieves a state of relaxed- $\bar{L}_{\mathcal{B}}(\lambda)$ -consistency.

*Remark 2.* When the filtering algorithm for  $\mathcal{B}$  only achieves a state of relaxed- $\bar{L}_{\mathcal{B}}(\lambda)$ -consistency, CP-based Lagrangian Relaxation is at most as effective as a filtering algorithm for the overall problem that achieves relaxed- $\bar{L}$ -consistency.

*Remark 3.* Due to symmetry in the argumentation, we can also deduce that for every variable  $x_j$  and value  $v \in \{l_j, \dots, u_j\}$  that can be filtered with respect to the relaxation  $\bar{L}$ , there exists a vector of Lagrangian multipliers that allows us to filter this value with respect to the constraint family  $\mathcal{A}$  only. This means that, when we consider the right Lagrangian multipliers, it is completely sufficient to perform cost-based filtering with respect to one substructure only!

*Remark 4.* The proof of Theorem 1 also tells us what the best filtering penalties are when our cost-based filtering algorithm for substructure  $\mathcal{B}$  achieves relaxed- $\bar{L}_{\mathcal{B}}(\lambda)$ -consistency. They are not the best multipliers for the constraints in  $\mathcal{A}$  in  $\bar{L}$ , but the best penalties for  $\mathcal{A}$  in  $\bar{L}[x_j = v]$ ! This is the reason why suboptimal multipliers can be more efficient for filtering than the optimal multipliers for the original problem.

Note that it is only the last observation that, for the first time, justifies the procedure of CP-based Lagrangian Relaxation that we proposed first in [23] and that consists in performing domain filtering not just for the optimal multipliers only, but for *all* Lagrangian multipliers that we encounter during the optimization of the Lagrangian dual. However, Remark 4 does not imply that the search for good Lagrangian multipliers should be terminated early, since an efficient tree search approach will always rely on the pruning power of good lower bound estimates. The remark just explains why it makes sense to perform cost-based filtering *during* the optimization of the Lagrangian dual and not just for the final, optimal or near-optimal multipliers only. Moreover, as we will see in the following two subsections, the observation that suboptimal Lagrangian multipliers can have stronger filtering abilities than the optimal penalties has severe consequences for the behavior of the entire method.

Another aspect is worth noting here<sup>3</sup>: The effect that a worse bound could lead to better filtering is actually not specific for Lagrangian relaxation. Suppose that we use a reduced-cost filtering algorithm for bounds  $L_1$  and  $L_2$ , i.e., we filter  $v$  from the domain of variable  $x_i$  iff  $L_j + \bar{c}_{iv}^j \geq B$  whereby  $B$  is an upper bound,  $1 \leq j \leq 2$ , and  $\bar{c}_{iv}^j$  denotes the reduced costs of setting  $x_i = v$  for relaxation  $j$ . You may want to think of  $L_1$  and  $L_2$  as emerging from two different, dual feasible solutions. Clearly, it is then possible that  $L_1 \geq L_2$  but  $\bar{c}_{iv}^1 < \bar{c}_{iv}^2$ . Then, it may very well be the case that  $L_2 + \bar{c}_{iv}^2 \geq B \geq L_1 + \bar{c}_{iv}^1$ , i.e. it may so happen that lower bound  $L_2$ , although providing a worse lower bound overall, allows the reduced-cost filtering of  $v$  from the domain of  $x_i$ , while the better bound  $L_1$  does not. Therefore, when we use for example a dual simplex algorithm to compute a lower bound, it makes perfect sense to apply reduced-cost filtering not only on the optimal simplex tableau, but also in every simplex iteration.

Note that, in the above discussion, we used the term “bound” in a very loose manner. It could really mean both: a specific lower bound that is usually associated with one dual feasible solution, or even a completely different bound, for example as represented by another relaxation. This being said, it is worth

---

<sup>3</sup> We owe this observation to an anonymous referee - thanks!

noting that, for CP-based Lagrangian Relaxation, also a dominated relaxation could result in more filtering. This stands in contrast to relaxed consistency where a dominant bounding routine (i.e. a bound that is always not worse than another) will always achieve at least as much filtering efficiency as a dominated bound. In that regard, the counter-intuitive effect of worse bounds pruning more is really caused by the heuristic nature of CP-based Lagrangian Relaxation and reduced-cost filtering.

## 4.2 Solving the Lagrangian Dual and Idempotence

When using CP-based Lagrangian Relaxation, after having shrunk the domain of the variables, the immediate re-application of the filtering algorithm may yield a further reduction of the domains. This effect is a result of Remark 4 and the fact that the algorithms used for the maximization of the Lagrangian dual – such as subgradient algorithms, bundle methods or the volume algorithm [3] – will in general proceed differently when the domains of the variables have changed. As a result, different Lagrangian multipliers and subproblems are investigated, which may very well result in a different filtering behavior. As a consequence, the filtering procedure as described is not idempotent [2].

Moreover, as mentioned in Section 3, it is not clear whether domain reduction should actually take place during the optimization of the Lagrangian dual. We are safe if we just mark those values that can be deleted from variable domains and postpone the actual reduction until the Lagrangian dual is solved. On the other hand, it may be also favorable to incorporate the new knowledge as early as possible. It is subject to further research to investigate how e.g. a subgradient search can cope with changing problems, and whether convergence can still be proven in such a scenario. A practical evaluation of how subgradient algorithms can cope with changing subproblems (for example by resetting the step-length factor) was published in [26].

## 4.3 Continuity

Since the filtering behavior of the reduction algorithm based on Lagrangian relaxation relies on the subproblems investigated during the optimization of the Lagrangian dual, we cannot be sure that our cost-based filtering algorithm exhibits a property that we call *continuity*:

**Definition 1.** *Let  $B$  denote an upper bound on the minimization problem  $L$ , let  $C$  denote the current choice point and denote with  $B_C$  the best lower bound and with  $B_C[x = v]$  the best bound achieved regarding the removal of  $v$  from the domain of  $x$  in the current choice point  $C$ . Consider some variable  $x$  and  $v$  in the domain of  $x$  that was not removed, i.e.  $\delta := B - B_C[x = v] > 0$ . Now assume that a primal heuristic finds a new upper bound  $\bar{B} \leq B - \delta$  next. We call a cost-based filtering algorithm continuous, if it is guaranteed that in every child node  $D$  of the current choice point  $C$  it is detected that  $v$  can be removed from the domain of  $x$ .*

In other words: A cost-based filtering algorithm is called continuous if and only if we can guarantee that a call to the filtering algorithm with smaller domains and improved upper bounds will remove at least all values that could have been deleted on the original domains if the improved upper bound had been known at that time.

As an implication of our discussion in the previous section, despite the fact that  $B_{\mathcal{D}} \geq B_{\mathcal{C}}$ , CP-based Lagrangian Relaxation is not a continuous filtering technique: Let  $\lambda \leq 0$  denote Lagrangian multipliers in  $\mathcal{C}$  that are optimal for the filtering of  $v$  from the domain of  $x$ , i.e.  $L_{\mathcal{B}}(\lambda)[x = v] = B_{\mathcal{C}}[x = v]$ . (Note that Remark 4 tells us that in general  $L_{\mathcal{B}}(\lambda) < B_{\mathcal{C}}$ !) Then we cannot be sure that, when performing problem reduction in  $\mathcal{D}$ , the algorithm optimizing the Lagrangian dual will investigate the Lagrangian multipliers  $\lambda$ . Thus, it may very well be the case that

$$B_{\mathcal{D}}[x = v] < \bar{B} < B_{\mathcal{C}}[x = v] < B.$$

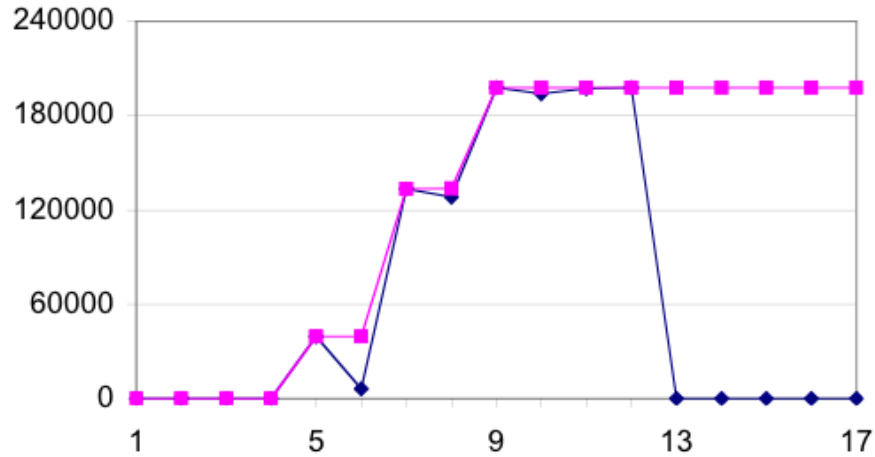
There is no obvious way how this problem could be overcome. However, our awareness of the problem can motivate strategies to strengthen the filtering abilities of our method. Assume for example, that the filtering algorithm for  $\mathcal{B}$  actually provides the values  $B_{\mathcal{C}}[x = v]$ , and for every variable-value-pair  $(x, v)$  we store the maximal value computed in all previous choice points. Then, as soon as the upper bound value  $B$  drops below this value,  $v$  can be removed from the domain of  $x$ . This procedure may be viewed as generation of local no-goods of the form:  $(B > \max_{\mathcal{C}} B_{\mathcal{C}}[x = v]) \vee (x \neq v)$ .

## 5 A Practical Experiment

After having studied the theoretical properties of CP-based Lagrangian Relaxation, we decided to conduct some experiments in order to see how the method behaves in practice. For this purpose, we consider the resource constrained shortest path problem (RCSSP) that consists in the conjunction of two shorter-path constraints [21]. We couple the two optimization constraints by CP-based Lagrangian Relaxation, using a specialized cutting plane algorithm [15] for the optimization of the Lagrangian dual.

In Figure 1 we show the run of the optimality proof for an instance that was taken from <http://www.mpi-sb.mpg.de/~mark/cnop>. It models the approximation of a one-dimensional curve by a piecewise-linear function with a constraint on the approximation error introduced and the objective to minimize the number of sampling points. The instance contains almost 200K directed edges.

In the first iterations (1–4), we can see how filtering is ineffective when only the isolated substructures are considered. In general, one would expect to see some slight filtering here, but for all our tests on the RCSSP we found that this is always very marginal when compared to the filtering that can be achieved by CP-based Lagrangian Relaxation. As the quality of the Lagrangian multiplier improves (iterations 5–9), we notice that filtering becomes more and more



**Fig. 1.** The graph shows the number of edges filtered over the iterations of the cutting plane algorithm optimizing the Lagrangian dual. The upper curve gives the cardinality of the set of all edges filtered until the current iteration, the lower curve gives the number of edges filtered in the current iteration only. Note that the upper curve is not the integral of the lower, since values can be filtered in multiple iterations; this happens in iterations 7 and 8, for instance.

effective. Then, in iteration 13, we suddenly notice that no filtering can be established anymore, even though we are close to finding the optimal multiplier in iteration 17. This clearly illustrates the fact that optimal multipliers need not have superior filtering abilities - as a matter of fact, the filtering achieved with optimal multipliers can be dramatically less. As we see, this is not an academic detail, but actually an effect that occurs for real-world instances.

While we observe a similar behavior for quite a few other RCSSP instances, of course the effect is not always that drastic. Often, we can also achieve very good filtering for the optimal multipliers. However, it is worth noting here that no one iteration alone is capable of filtering all edges. Iterations 9–12 come close to achieving maximum filtering effectiveness, but, as the exact numbers show, in none of them all edges are filtered that are detected throughout the optimization of the Lagrangian dual. We believe that the fact that these iterations are so effective at all is more an expression of the simple structure of the RCSSP. In general, multipliers for the filtering of different values will certainly differ as well.

## 6 Conclusion

We investigated the theoretical foundations of CP-based Lagrangian Relaxation. An example showed that suboptimal Lagrangian multipliers can have stronger filtering abilities than optimal penalties. We were able to explain this counter-intuitive observation by proving the simple but central Theorem of Filtering Penalties. It allowed us to derive several properties of CP-based Lagrangian Relaxation: 1. When cost-based filtering for the optimization substructures is based on linear continuous bounds, CP-based Lagrangian Relaxation is at most as effective as a filtering algorithm that achieves relaxed- $\bar{L}$ -consistency for the entire problem. 2. If we traverse the right Lagrangian multipliers during the optimization of the Lagrangian dual and filtering is based on linear continuous bounds, then filtering with respect to one substructure alone is as effective as filtering with respect to all substructures. 3. CP-based Lagrangian Relaxation is neither an idempotent nor a continuous propagation technique.

Our hope is that the theoretical understanding of the technique will help to engineer superior solution approaches for composed combinatorial optimization problems. We have seen already that the theoretical discussion can stimulate new ideas such as the addition of local no-goods. An important practical question regards the time when the actual filtering effects are executed and whether this can be done during the optimization of the Lagrangian dual. Some work in the past has focused on improving the lower bounds during the solution of the Lagrangian dual. Our work shows that this does not necessarily imply an improved filtering behavior, and careful testing is required to evaluate the practical benefits of doing so.

Finally, we have seen that it can be as efficient when filtering with respect to just one substructure only as it is when applying filtering algorithms to all substructures in the problem. Our previous practical experience confirms this theoretical finding. For capacitated network design for example, it did not pay off to filter with respect to both the shortest path and the Knapsack substructures [26]. Moreover, in the context of automatic recording the application of Knapsack filtering on top of weighted stable set filtering only paid off marginally [25]. Note also that for this problem the number of Lagrangian subproblems that had to be investigated to optimize the Lagrangian dual was very low, which decreases our chances to traverse the good filtering penalties. Therefore, attempts to integrate several filtering algorithms in a Lagrangian fashion must be viewed very critically: Filtering more than one substructure may be a pure waste of time, especially when filtering is based on linear bounds and when many Lagrangian subproblems are investigated during the solution of the Lagrangian dual.

## 7 Acknowledgment

We would like to thank Robert Wright for conducting many numerical experiments on the RCSSP and for providing Figure 1.

## References

1. R.K. Ahuja, T.L. Magnati, and J.B. Orlin. *Network Flows*. Prentice Hall, 1993.
2. K. R. Apt. The Rough Guide to Constraint Propagation. *5th International Conference on Principles and Practice of Constraint Programming (CP)*, LNCS 1713:1–23, 1999.
3. F. Barahona and R. Anbil. The Volume Algorithm: producing primal solutions with a subgradient algorithm. *Mathematical Programming*, 87:385–399, 2000.
4. Y. Caseau and F. Laburthe. Solving Small TSPs with Constraints. *14th International Conference on Logic Programming (ICLP)*, pp. 316–330, The MIT Press, 1997.
5. H. Crowder. Computational improvements for subgradient optimization. *Symposia Mathematica*, XIX:357–372, 1976.
6. H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resource. *Operations Research*, 11:399–417, 1963.
7. T. Fahle and M. Sellmann. Cost-Based Filtering for the Constrained Knapsack Problem. *Annals of Operations Research*, 115:73–93, 2002.
8. F. Focacci, A. Lodi, and M. Milano. Cutting Planes in Constraint Programming: an hybrid approach. *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP)*, Springer LNCS 1894:187–201, 2000.
9. F. Focacci, A. Lodi, and M. Milano. Cost-Based Domain Filtering. *5th International Conference on Principles and Practice of Constraint Programming (CP)*, LNCS 1713:189–203, 1999.
10. F. Focacci, A. Lodi, and M. Milano. Solving TSP through the Integration of OR and CP Techniques. *Workshop on Large Scale Combinatorial Optimization and Constraints*, Electronic Notes in Discrete Mathematics, 1998.
11. A. Frangioni. A Bundle type Dual-ascent Approach to Linear Multi-Commodity Min Cost Flow Problems. *Technical Report*, Dipartimento di Informatica, Università di Pisa, TR-96-01, 1996.
12. A. Frangioni. Dual Ascent Methods and Multicommodity Flow Problems. *Doctoral Thesis*, Dipartimento di Informatica, Università di Pisa, TD-97-05, 1997.
13. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.
14. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.
15. J.E. Kelley. The Cutting Plane Method for Solving Convex Programs. *Journal of the SIAM*, 8:703–712, 1960.
16. G. Kliewer, M. Sellmann, and A. Koberstein. Solving the capacitated network design problem in parallel. *3rd meeting of the PAREO Euro working group on Parallel Processing in Operations Research (PAREO)*, 2002.
17. S. Martello and P. Toth. An exact algorithm for the Two-Constraint 0-1 Knapsack Problem. *Operations Research*, 51:826–835, 2003.
18. G. Ottosson and E.S. Thorsteinsson. Linear Relaxation and Reduced-Cost Based Propagation of Continuous Variable Subscripts. *2nd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, Paderborn Center for Parallel Computing, Technical Report tr-001-2000:129–138, 2000.
19. M. Sellmann. The Practice of Approximated Consistency for Knapsack Constraints. *National Conference on Artificial Intelligence (AAAI)*, to appear, 2004.

20. M. Sellmann. Approximated Consistency for Knapsack Constraints. *Principles and Practice of Constraint Programming (CP)*, Springer LNCS 2833: 679–693, 2003.
21. M. Sellmann. Cost-Based Filtering for Shorter Path Constraints. *Principles and Practice of Constraint Programming (CP)*, Springer LNCS 2833: 694–708, 2003.
22. M. Sellmann. Pruning Techniques in Constraint Programming and Combinatorial Optimization. *Ph.D. Thesis*, University of Paderborn, 2002.
23. M. Sellmann and T. Fahle. CP-Based Lagrangian Relaxation for a Multimedia Application. *3rd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, pp. 1–14, 2001.
24. M. Sellmann and T. Fahle. Coupling Variable Fixing Algorithms for the Automatic Recording Problem. *9th Annual European Symposium on Algorithms (ESA)*, LNCS 2161:134–145, 2001.
25. M. Sellmann and T. Fahle. Constraint Programming Based Lagrangian Relaxation for the Automatic Recording Problem. *Annals of Operations Research*, 118:17–33, 2003.
26. M. Sellmann, G. Kliewer, and A. Koberstein. Lagrangian Cardinality Cuts and Variable Fixing for Capacitated Network Design. *10th Annual European Symposium on Algorithms (ESA)*, LNCS 2461:845–858, 2002.
27. M. Trick. A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints. *3rd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, pp. 113–124, 2001.